C로 알아보는
소켓 프로그래밍

이현환(NOON)
haonun@gmail.com
http://noon.tistory.com

Hacking Study Grup E.Y.E

목차

서론

본론

결론

서론

현재의 많은 프로그램 , 프로젝트 들은 네트워크를 기반으로 한 프로그램들이 많습니다.
또한 최근에 활발한 발전을 이룩한 모바일 , 웹 기반의 어플리케이션들도 대부분 네트워크를 전재로 움직입니다. 이만큼 네트워크는 삶에 깊이 침투한 분야입니다.
우리그룹도 취약점을 찾고 , 또한 침투테스트를 하기 위해선 네트워크 프로그래밍의 중요성을 인식해야 할 것 같습니다. 네트워크 프로그래밍의 기본인 소켓프로그래밍을 배움으로서 우리가 사용해야 할 툴이나 프로그램들을 직접 개발할수 있는 환경이 되길 바랩니다.

본론

네트워크란

TCP/IP

TCP/IP는 **인터넷의 기본적인 통신 프로토콜**로서, 인트라넷이나 엑스트라넷과 같은 사설 망에서도 사용된다. 사용자가 인터넷에 접속하기 위해 자신의 컴퓨터를 설정할때 TCP/IP 프로그램이 설치되며, 이를 통하여 역시 같은 TCP/IP 프로토콜을 쓰고 있는 다른 컴퓨터 사용자와 메시지를 주고받거나, 또는 정보를 얻을 수 있게된다.

TCP/IP는 2개의 계층으로 이루어진 프로그램이다. 상위계층인 TCP는 메시지나 파일들을 좀더 작은 패킷으로 나누어 인터넷을 통해 전송하는 일과, 수신된 패킷들을 원래의 메시지로 재조립하는 일을 담당한다. 하위계층, 즉 IP는 각 패킷의 주소 부분을 처리함으로써, 패킷들이 목적지에 정확하게 도달할 수 있게 한다. 네트웍 상의 각 게이트웨이는 메시지를 어느 곳으로 전달해야 할지를 알기 위해, 메시지의 주소를 확인한다. 한 메시지가 여러 개의 패킷으로 나뉘어진 경우 각 패킷들은 서로 다른 경로를 통해 전달될 수 있으며, 그것들은 최종 목적지에서 재조립된다.

TCP/IP는 통신하는데 있어 클라이언트/서버 모델을 사용하는데, 컴퓨터 사용자(클라이언트)의 요구에 대응하여, 네트웍 상의 다른 컴퓨터(서버)가 웹 페이지를 보내는 식의 서비스를 제공한다. TCP/IP는 본래 점대점(点對点) 통신을 하는데, 이는 각 통신이 네트웍 상의 한 점(또는 호스트 컴퓨터)으로부터 시작되어, 다른 점 또는 호스트 컴퓨터로 전달된다는 것을 의미한다.

TCP/IP와 TCP/IP를 이용하는 상위계층의 응용프로그램들은 모두 "커넥션리스(connectionless)"라고 불리는데, 이는 각 클라이언트의 요구가 이전에 했던 어떠한 요구와도 무관한 새로운 요구로 간주된다는 것을 의미한다 (일상적인 전화통화가 통화시간 내내 지속적으로 연결되어 있어야 하는 것과는 다르다). 커넥션리스는 네트웍을 독점하지 않으므로, 모든 사람들이 그 경로를 끊임없이 공동으로 사용할 수 있게 한다 (사실 TCP 계층 그 자체는 어떤 한 메시지가 관계되어 있는 한 커넥션리스가 아니라는데 유의해야 한다. TCP 접속은 어떤 한 메시지에 속하는 모든 패킷들이 수신될 때까지 계속 유지된다).

많은 인터넷 사용자들이 TCP/IP를 이용하는 상위계층 응용프로토콜에 대해서는 잘 알고 있다. 이러한 상위계층 프로토콜에는 웹서비스에 사용되는 HTTP를 비롯하여, 멀리 떨어져 있는 원격지의 컴퓨터에 로그온할 수 있게 해주는 Telnet, 그리고 파일전송에 사용되는 FTP와 메일 전송에 사용되는 SMTP 등이 있다. 이러한 프로토콜들은 종종 TCP/IP와 함께 패키지로 일괄 판매된다.

PC 사용자들은 보통 인터넷에 접속하기 위해 SLIP이나 PPP 프로토콜을 사용한다. 이러한 프로토콜들은 다이얼업 전화접속을 통해 접속서비스사업자의 모뎀으로 보내질 수 있도록 IP 패킷들을 캡슐화한다.

TCP/IP와 관련이 있는 프로토콜로 UDP가 있는데, 이것은 특별한 목적을 위해 TCP 대신에 사용되는 것이다. 라우팅 정보를 교환하기 위해 네트웍 호스트 컴퓨터에 의해 사용되는 프로토콜에는 ICMP, IGP, EGP, 그리고 BGP 등이 있다.

소켓프로그래밍

함수설명

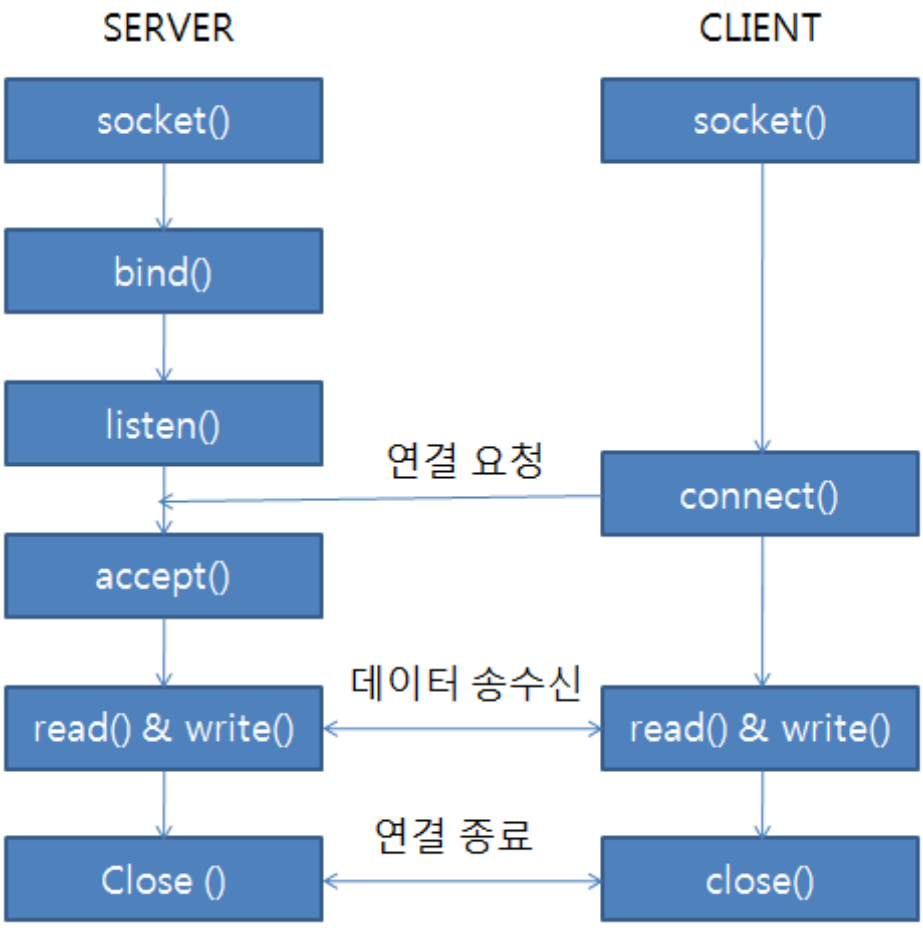
socket : 소켓생성
bind : 주소부여
listen : 대기
accept : 연결확인 및 수신
connect : 접속
read : 읽기
write : 쓰기

서버

1. 소켓 생성(socket 함수)
2. 생성된 소켓에 인터넷 주소 부여(bind 함수)
3. 데이터 수신 대기(listen 함수)
4. 데이터 수신(accept 함수)
5. 데이터 읽기(read 함수)
6. 데이터 쓰기(write 함수)
7. 3번으로 돌아간다(listen 함수)

클라이언트

1. 소켓 생성(socket 함수)
2. 서버에 연결(connect 함수)
3. 데이터 쓰기(write 함수)
4. 데이터 읽기(read 함수)
5. 연결 종료(close 함수)



C로 알아보기

서버

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>

void error_handling(char *message);

int main(int argc, char **argv)
{
	int sock;
	struct sockaddr_in serv_addr;
	char message[30];
	int str_len;
	int ending = 0;

	if(argc!=3)
	{
		printf("Usage : %s <IP> <port>₩n", argv[0]);
		exit(1);
	}

	while(!ending)
	{
		sock = socket(PF_INET, SOCK_STREAM, 0); //서버 접속을 위한 소켓 생성
		if(sock == -1)
		{
			error_handling("socket() error");
		}

		memset(&serv_addr, 0, sizeof(serv_addr)); //메모리 초기화

		serv_addr.sin_family = AF_INET; //인터넷 주소 체계 저장
		serv_addr.sin_addr.s_addr = inet_addr(argv[1]); //
		serv_addr.sin_port = htons(atoi(argv[2]));

		if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1) //서버로 연결
요청
			error_handling("connect() error");

		str_len = read(sock, message, sizeof(message) -1); //데이터 수신

		if(str_len == -1)
			error_handling("read() error!");

		message[str_len] = 0;
```

```
                printf("Message from server : %s ₩n", message);
                if(message[0] == 0x1b) ending = 1;

                close(sock); //연결 종료
        }
        return 0;
}


void error_handling(char *message)
{
        fputs(message, stderr);
        fputc('₩n', stderr);
        exit(1);
}
```

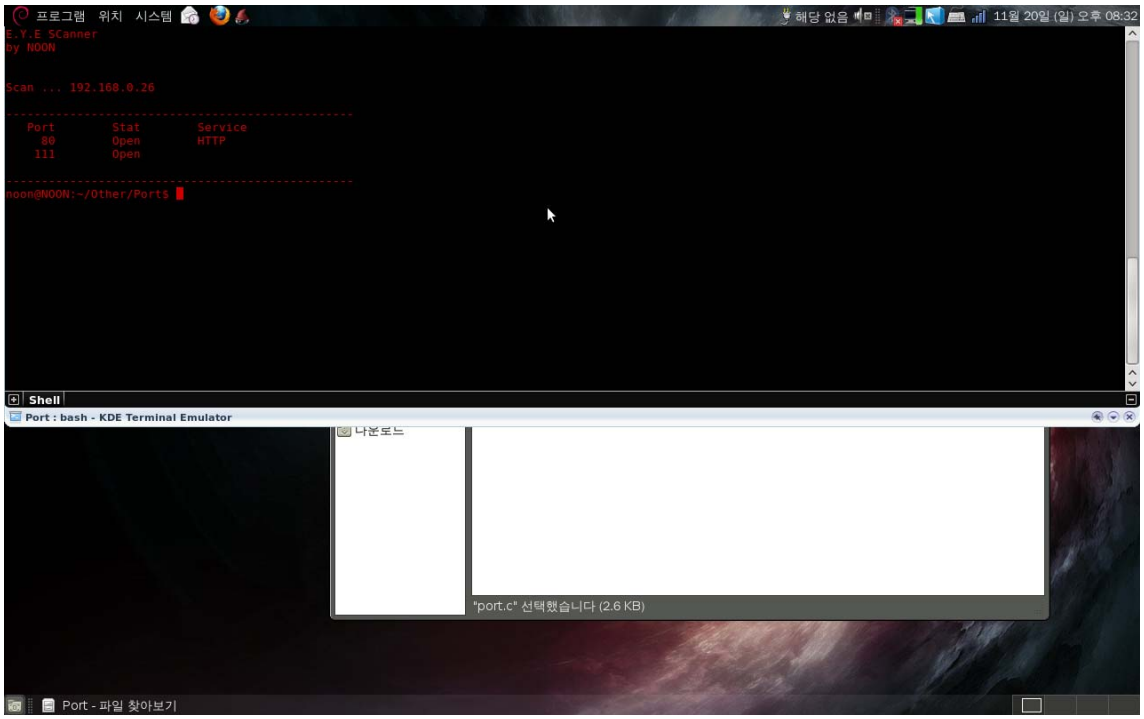
클라이언트

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>

void error_handling(char *message);

int main(int argc, char **argv)
{
        int i;
        int serv_sock;
        int clnt_sock;
        struct sockaddr_in serv_addr;
        struct sockaddr_in clnt_addr;
        int clnt_addr_size;
        char message[256];
        int ending = 0;

        if(argc != 2)
        {
                printf("Usage : %s <port>₩n", argv[0]);
                exit(1);
        }
        serv_sock = socket(PF_INET, SOCK_STREAM, 0); //서버 소켓 생성

        if(serv_sock == -1)
                error_handling("socket() error");

        memset(&serv_addr, 0, sizeof(serv_addr));

        serv_addr.sin_family = AF_INET; //Internet protocol
```

```
	serv_addr.sin_addr.s_addr = htonl(INADDR_ANY); //Ip address
	serv_addr.sin_port = htons(atoi(argv[1])); //Port

	if(bind(serv_sock, (struct socketaddr*) &serv_addr, sizeof(serv_addr)) == -1) // 소켓에 주소 할당
		error_handling("bind() error");

	if(listen(serv_sock, 5) == -1) //listen stat 돌입
		error_handling("listen() error");

	clnt_addr_size = sizeof(clnt_addr);

	while(!ending)
	{
		clnt_sock=accept(serv_sock, (struct sockaddr*) &clnt_addr, &clnt_addr_size); //연결 요
청이 들어오면 수락

		if(clnt_sock == -1)
			error_handling("accept() error");

		scanf("%s", message);
		write(clnt_sock, message, sizeof(message)); //전송할 데이터 쓰기

		if(message[0] == 0x1b) ending = 1;
	}
	close(clnt_sock);

	return 0;
}


void error_handling(char *message)
{
	fputs(message, stderr);
	fputc('₩n', stderr);
	exit(1);
}
```

응용

위에서 배운 소켓프로그래밍 기술로 간단한 포트스캐너를 구현하였습니다.
포트스캐너는 어떠한 네트워크에 대해 많은 정보를 취득할수 있는 도구입니다.
대표적인 포트스캐너로는 nmap 이 있습니다. (관련 http://noon.tistory.com/689)



현재는 간단한 예제로 사용하기 위해 오픈되어 있는 포트와 서비스의 종류를 탐지할수 있도록 설계해 보았습니다. 현재 Hidden Eye라는 포트 및 취약점 스캐너를 만들어보면서 간단하게도 이렇게 구성할수 있다는걸 배웠기에 이 내용을 공유합니다.

첨부로 소스코드 올려습니다.
밑의 포트스캐너는 리눅스에서 사용하실 수 있습니다.

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#include <errno.h>
int searcheye(char *hostaddr, int cport, int socktype);
char service[20]; //서비스 확인
int main(int argc, char *argv[])// 프로그램 싱행 인자값
{           int cport; // count Port
                                if (argc < 2) {
            printf("port 'hostname' [not Option]₩n");
            return 0;
            }
system("clear");
```

```
printf("E.Y.E SCanner₩nby NOON₩n");
printf("₩n₩nScan ... %s₩n",argv[1]);
printf("₩n-------------------------------------------------₩n");
printf("    Port          Stat          Service₩n");
                    for (cport = 1;cport < 1001;cport++)   //확인포트
          {
                              if (searcheye(argv[1], cport, SOCK_STREAM) == 0) //연결값 출력
                    {
                    printf("   %5d          Open       %7s₩n", cport,service);
                    memset(service,0x00, sizeof(service)); //문자열 초기화
                    }
          }
printf("₩n-------------------------------------------------₩n");
return 0;
}int searcheye(char *hostaddr, int cport, int socktype)
{         int sockfd; // 소켓
          struct hostent *he;
                    struct sockaddr_in destaddr;//socket addr 를 좀더 편하게 넣기위한
          int pok; //반환값
          if ((he = gethostbyname(hostaddr)) == NULL) {
          herror("gethostbyname"); //겟바이 호스트
return 0;
}sockfd = socket(AF_INET, socktype, 0); //sockfd 소켓으로 사용함
destaddr.sin_family = AF_INET; // AF_INET    TCPIP
destaddr.sin_addr = *((struct in_addr *)he->h_addr); // IP
destaddr.sin_port = htons(cport); //포트
bzero(&(destaddr.sin_zero), 8);
// 연결
pok = connect(sockfd, (struct sockaddr *)&destaddr, sizeof(struct sockaddr)); //실패시 -1반환
close(sockfd); //연결종료
if (pok == -1) //반환 값으로 판단
          {
          return -1; //닫혀있으면 -1
          }
switch(cport) //아니면 서비스 탐지
{         case 21:  strcpy(service,"FTP"); break;
          case 22:  strcpy(service,"SSH"); break;
          case 23:  strcpy(service,"Telnet"); break;
          case 25:  strcpy(service,"SMTP"); break;
          case 79:  strcpy(service,"Finger"); break;
          case 80:  strcpy(service,"HTTP"); break;
          case 119:strcpy(service,"NNTP"); break;
          case 137:strcpy(service,"NetBios"); break;
          case 138:strcpy(service,"NetBios"); break;
          case 139:strcpy(service,"NetBios"); break;
}return 0; // 0 반환
}
```

결론

짧은시간에 급히 써서 발표하게된 글이라 허술하고 내용이 많이 부실합니다. 그러나 이 문서를 통하여 소켓이라는것에 흥미라도 느껴주셧으면 고맙겠습니다.
침투테스트 , 서버점검등 여러 가지 일들을 수행하기 위해선 그에 맞는 툴과 프로그램들이 필요한데 그걸 직접 개발할수 있도록 기초를 쌓는 시간이 되셨으면 좋겠습니다.

감사합니다.

다른 자료들은 따로 첨부하겠습니다.